

Contents

1	Routine/Function Prologues	2
1.1	Fortran: Module Interface geosopendap_module.F90 (Source File: geosopendap_module.F90)	2
1.1.1	reset_geos_filepaths (Source File: geosopendap_module.F90)	3
1.1.2	init_geos_vars (Source File: geosopendap_module.F90)	3
1.1.3	def_kgds (Source File: geosopendap_module.F90)	5
1.1.4	init_geos_ref_date (Source File: geosopendap_module.F90)	5
1.1.5	get_geos_index (Source File: geosopendap_module.F90)	5
1.1.6	set_geos_lat (Source File: geosopendap_module.F90)	6

1 Routine/Function Prologues

1.1 Fortran: Module Interface geosopendap_module.F90 (Source File: geosopendap_module.F90)

This module contains routines needed to initialize and control variables required for the execution of GDS-based I/O specific to GEOS forcing routines

REVISION HISTORY:

10 Jul 2003; James Geiger Initial Specification
 22 Dec 2003; Sujay Kumar Separated geos specific routines from the main module

INTERFACE:

```
module geosopendap_module
#if ( defined OPENDAP )
```

USES:

```
use geosdomain_module, only : geosdrv
use ESMF_TimeMgmtMod
use lisdrv_module,      only : lis, grid, gindex
use grid_spmMod,       only : gdi, gdisp
use tile_spmMod,       only : di_array
use spmMod
use opendap_module,    only : opendap_data_prefix, ciam
```

```
implicit none
```

```
type(esmf_date), save :: geos_ref_date !Reference date for GEOS forcing data
integer                :: geos_slat    !Southern latitude boundary for each subdomain (for
integer                :: geos_nlat    !Northern latitude boundary for each subdomain (for
integer                :: geos_nc      !Number of columns for each subdomain (for GEOS)
integer                :: geos_nr      !Number of rows for each subdomain (for GEOS)
integer                :: input_slat   !Southern latitude boundary for the native domain
integer                :: input_nlat   !Northern latitude boundary for the native domain
character*4            :: cgeos_slat   !character representation of geos$-$slat
character*4            :: cgeos_nlat   !character representation of geos$-$nlat
integer                :: grid_offset  !Global to local grid mapping offset
integer                :: fnroffset
```

```
contains
```

```
%////////////////////////////////////
```

```
\mbox{\hrulefill\
```

```
\subsubsection{opendap\_geos\_init (Source File: geosopendap\_module.F90)}
```

Initializes the GEOS-GDS variables

```
\bigskip{\sf INTERFACE:}
\begin{verbatim}  subroutine opendap_geos_init()
```

CONTENTS:

```
  call init_geos_vars()
  call reset_geos_filepaths()
```

1.1.1 reset_geos_filepaths (Source File: geosopendap_module.F90)

Resets input data filenames for GEOS forcing for execution through GDS

INTERFACE:

```
  subroutine reset_geos_filepaths()
```

CONTENTS:

```
  geosdrv%geosdir      = trim(opendap_data_prefix)//'/ '// &
                        trim(adjustl(ciam))//'/ '//geosdrv%geosdir
```

1.1.2 init_geos_vars (Source File: geosopendap_module.F90)

Computes domain decomposition for native as well as interpolated domains for input GEOS forcing

INTERFACE:

```
  subroutine init_geos_vars()
```

CONTENTS:

```
  select case (lis%d%domain)
  case(1)
    print*, 'Error!  Cannot handle nldas.'
    stop 999
  case(2) ! 1 deg
    res = 1000
    center = 500
    bottom = -59500
  case(3)
    print*, 'Error!  Cannot handle 2x2.5.'
    stop 999
  case(4) ! 1/4 deg
    res = 250
```

```
        center = 875
        bottom = -59875
case(5) ! 1/2 deg
        res = 500
        center = 750
        bottom = -59750
case(6) ! 5km
        res = 50
        center = 975
        bottom = -59975
case(7) ! 5km
        res = 50
        center = 975
        bottom = -59975
case(8) ! 1km
        res = 10
        center = 995
        bottom = -59995
case DEFAULT
        print*, "Select domain size (1,2,3,4,5,6,7,8)"
        stop 999
end select

call set_geos_lat(geos_slat,geos_nlat,input_slat,input_nlat)

lis%d%ngrid = gdi(iam)
lis%d%nch   = di_array(iam)

geos_nc     = 360
geos_nr     = ( geos_nlat - geos_slat + 1 )

fnroffset = geos_slat - 1
grid_offset = tile(1)%index-1

write(cgeos_slat, '(i4)') geos_slat
write(cgeos_nlat, '(i4)') geos_nlat

print*, 'DBG: geos_init -- geos_slat', geos_slat, ' (', iam, ')'
print*, 'DBG: geos_init -- geos_nlat', geos_nlat, ' (', iam, ')'
print*, 'DBG: geos_init -- ngrid', lis%d%ngrid, ' (', iam, ')'
print*, 'DBG: geos_init -- glbngrid', lis%d%glbngrid, ' (', iam, ')'
print*, 'DBG: geos_init -- ncold', geosdrv%ncold, ' (', iam, ')'
print*, 'DBG: geos_init -- nrold', geosdrv%nrold, ' (', iam, ')'
print*, 'DBG: geos_init -- lnc', lis%d%lnc, ' (', iam, ')'
print*, 'DBG: geos_init -- lnr', lis%d%lnr, ' (', iam, ')'
print*, 'DBG: geos_init -- fnroffset', fnroffset, ' (', iam, ')'
print*, 'DBG: geos_init -- grid_offset', grid_offset, ' (', iam, ')'
print*, 'DBG: geos_init -- cgeos_slat ', cgeos_slat, ' (', iam, ')'

```

```
print*, 'DBG: geos_init -- cgeos_nlat ', cgeos_nlat, ' (', iam, ')'  
print*, 'DBG: geos_init -- geos_nc', geos_nc, ' (', iam, ')'  
print*, 'DBG: geos_init -- geos_nr', geos_nr, ' (', iam, ')'
```

1.1.3 def_kgds (Source File: geosopendap_module.F90)

Initializes the kgds array for GEOS-GDS runs.

INTERFACE:

```
subroutine def_kgds(kgdsi)
```

CONTENTS:

```
kgdsi(1) = 0  
kgdsi(2) = geos_nc  
kgdsi(3) = geos_nr  
kgdsi(4) = input_slat  
kgdsi(7) = input_nlat  
kgdsi(5) = -180000  
kgdsi(6) = 128  
kgdsi(8) = 179000  
kgdsi(9) = 1000  
kgdsi(10) = 1000  
kgdsi(20) = 255
```

1.1.4 init_geos_ref_date (Source File: geosopendap_module.F90)

Initializes the reference date for GEOS forcing data

INTERFACE:

```
subroutine init_geos_ref_date()
```

CONTENTS:

```
geos_ref_date = esmf_dateinit(esmf_no_leap, ref_ymd, ref_tod, rc)
```

1.1.5 get_geos_index (Source File: geosopendap_module.F90)

Computes the time-based index for GEOS forcing data

INTERFACE:

```
function get_geos_index(offset)
  implicit none
```

INPUT PARAMETERS:

```
integer, intent(in) :: offset ! offset from current date in hours
```

CONTENTS:

```
if ( ref_data_uninit ) then
  print*, 'DBG: get_geos_index -- initializing ref date', ' (', iam, ')'
  call init_geos_ref_date()
  ref_data_uninit = .false.
endif
ymd = ( lis%t%yr * 10000 ) + ( lis%t%mo * 100 ) + lis%t%da
tod = ( lis%t%hr * 3600 ) + ( lis%t%mn * 60 ) + lis%t%ss
current_date = esmf_dateinit(esmf_no_leap, ymd, tod, rc)

diff = esmf_timeinit()
call esmf_datediff(current_date, geos_ref_date, diff, islater, rc)
call esmf_timeget(diff, ndays, nsecs, rc)

get_geos_index = ( (ndays * 24) + (nsecs / 3600) + offset ) / 3 + 1
print*, 'DBG: get_geos_index -- get_geos_index = ', get_geos_index, &
  ' (', iam, ')'
```

1.1.6 set_geos_lat (Source File: geosopendap_module.F90)

Computes the latitudes of the decomposed domain for GEOS forcing data

INTERFACE:

```
subroutine set_geos_lat(slat, nlat, islat, inlat)
  implicit none
```

OUTPUT PARAMETERS:

```
integer, intent(out) :: slat, nlat, islat, inlat
```

CONTENTS:

```
!-----
! Set southern latitude index
!-----
print*, 'DBG: set_geos_lat -- grid(1)%lat', grid(1)%lat, ' (', iam, ')'
lat = grid(1)%lat
if ( lat < 0.0 ) then
  slat = int(lat) - 1 ! E.g. map -54.5 \to -55
else
```

```
        slat = int(lat)      ! E.g. map 40.5 \to 40
    endif
    if ( slat < -90 ) then
        print*, 'ERR: set_geos_lat -- Setting slat = -90', &
            ' (' , iam, ') '
        slat = -90
    endif
!-----
! Set southern latitude boundary
!-----
        islat = 1000 * slat
!-----
! Set northern latitude index
!-----
        print*, 'DBG: set_geos_lat -- grid(gdi(iam))', &
            gdi(iam), grid(gdi(iam))%lat, ' (' , iam, ') '
        lat = grid(gdi(iam))%lat
        if ( lat < 0.0 ) then
            nlat = int(lat)      ! E.g. map -10.5 \to -10
        else
            nlat = int(lat) + 1 ! E.g. map 10.5 \to 11
        endif
        if ( nlat > 90 ) then
            print*, 'ERR: set_geos_lat -- Setting nlat = 90', &
                ' (' , iam, ') '
            nlat = 90
        endif
!-----
! Set northern latitude boundary
!-----
        inlat = 1000 * nlat

        slat = slat + 90 + 1 ! map [-90,90] \to [1,181]
        nlat = nlat + 90 + 1 ! map [-90,90] \to [1,181]
        slat = 1
        nlat = 181
        islat = -90000
        inlat = 90000
```